Animation Project in C++ using Polymorphic Inheritance and RTTI

Due Time: 23.59, Sat 6 January 2018 **Earnings:** 9% of your final grade

NOTE: Plan to finish a few days early to avoid last minute hardware/software holdups for which no allowance is given.
NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.

Purpose: This is a development of assignment 2. It works in a very similar way but with the addition of two new classes SystemMemoryDisplay and GPUMemoryDisplay, that are derived from Display which has now become an abstract base class. Like assignment 2, it is a console application that holds the Frames of an Animation application (there is no actual animation graphics in the assignment) in a `forward_list` class template of unspecified length in dynamic memory. Each frame now holds a vector of Display* pointers that are the addresses of the Display objects that are actually SystemMemoryDisplay or GPUMemoryDisplay objects. Polymorphic inheritance ensures that any Display* in the vector will call the correct overridden polymorphic function (Buffersize() in this case, together with the destructor) for the actual object it points to. Also a SystemMemoryDisplay object uses more memory than a GPUMemoryDisplay object because they reside in different memory locations in the computer: in system memory or GPU local memory respectively. A GPUMemoryDisplay object also has a shader file that is a small fragment of code that executes for each pixel in the display buffer.
Therefore you will use an expression like

```
displays[i]->BufferSize()
```

to cause one of the Display* in the Frame vector to calculate the buffer size of the SystemMemoryDisplay or GPUMemoryDisplay it points to without needing to identify what type of Display it really is. The polymorphic function that actually executes is the correct one for the object type, selected through the virtual function table of the object.
In places where you need to identify the type of Display (SystemMemoryDisplay or GPUMemoryDisplay), ***but not where polymorphism is appropriate***, use `dynamic_cast<>`.
To focus on the polymorphic aspects of this final assignment, some of the overloaded operators used in Assignment 2 have been removed.

Part of the code is shown on the next page; it is also on the Web Site in a text file that you can copy and paste. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no new global variables or functions, no new classes, no macros, no defines and no statics**. Your task is to implement, using C++, only the Animation, Frame and Display class member functions and the global insertion operators and not add any new ones. Everything you write and submit is in the files: Animation.cpp, Frame.cpp and Display.cpp.

The Animation is a series of Frames held in a `forward_list`. Each Frame holds its list of Display* in a `vector`. There are now two types of Display, as detailed above, both subclasses of the abstract base class Display. Each Display object contains its display time which is set by the user. You can:

- Add a new Frame to the Animation at a position in the forward list selected by the user
- Delete all the Frames in the Animation
- Run the Animation to show the list of Display details of each Frame one after another at the display intervals specified by the user when the Display was entered – note that the output counts up the seconds using a timer as was done in assignment 0 onwards.
- Quit

An example of the output of the running application is given at the end. Yours must work identically and produce identical output.

Note the following:

- dynamic memory management is done with new and delete (not malloc, realloc or free)
- input and output is done with cin and cout
- there is no unused dynamic memory at any time
- string objects are used in the Animation and Frame classes to hold strings (a char array is still used in the Display class)
- Release of dynamically allocated memory is done in destructors so there is no resource leak (or you lose 30%).

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:
1. you change the supplied code in any way at all (not a single character) - no code added or removed, no macros, no defines, no statics and no additional classes, global functions or variables,
2. it fails to build in Visual Studio 2015,
3. It crashes in normal operation,
4. it doesn't produce the example output.

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to add the implementation of the class member functions.

What to Submit : Use Blackboard to submit this assignment as a plain zip file (**not** RAR or 7-Zip or 9 Zip) containing only Animation.cpp, Frame.cpp and Display.cpp. The name of the zipped folder **must** contain your name as a prefix so that I can identify it, for example using my name the file would be tyleraAss3CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header.

Before you submit the code,

- check that it builds and executes in Visual Studio 2015 as you expect - if it doesn't build for me, for whatever reason, you get a deduction of at least 60%.
- make sure you have submitted the correct file – if I cannot build it because the file is wrong or missing from the zip, even if it's an honest mistake, you get 0.

It cannot be late – no time. Don't send me files as an email attachment – they will get 0.

Supplied code (also in a text file you can copy and paste on BlackBoard). Don't change it.

```
#pragma once
// Display.h

class Display
{
protected:
          int pixel_x;
          int pixel_y;
          int duration;
          char* name;
public:
          Display(int x, int y, int duration, char* name);
          Display(const Display&);
          virtual ~Display()
          {
                    if(name)
                              delete[]name;
          }
          virtual int BufferSize() = 0;
          friend ostream& operator<<(ostream&, Display&);
};
```

```
#pragma once
//GPUMemoryDisplay.h

class GPUMemoryDisplay : public Display
{
          string shader;
public:
          GPUMemoryDisplay(int x, int y, int duration, char* name, string shader) :Display(x, y, duration, name),
shader(shader) {};
          GPUMemoryDisplay(const GPUMemoryDisplay& RGPUMD) :shader(RGPUMD.shader), Display(RGPUMD) {}
          string GetShader() { return shader; }
          int BufferSize() { return pixel_x*pixel_y * sizeof(float); }
};
```

```
#pragma once
// SystemMemoryDisplay.h

class  SystemMemoryDisplay : public Display
{
public:
          SystemMemoryDisplay(int x, int y, int duration, char* name) :Display(x, y, duration, name) {};
          SystemMemoryDisplay(const SystemMemoryDisplay& RGMD) :Display(RGMD) {}
          int BufferSize(){return pixel_x*pixel_y * sizeof(double);}
};
```

```
// Frame.h
#pragma once

class Frame
{
          string fileName;
          vector<Display*> displays;
public:
          Frame(string s, vector<Display*> d) :fileName(s), displays(d){}
          Frame(const Frame&);
          ~Frame()
          {
                    vector<Display*>::iterator it;
                    for (it = displays.begin();it != displays.end();it++)
                              delete *it;
          }
          friend ostream& operator<<(ostream&, Frame&);
};
```

```
//Animation.h
#pragma once

class Animation
{
          string name;
          forward_list<Frame> frames;
public:
          Animation(string s): name(s){}
          void InsertFrame();
          void DeleteFrames();
          friend ostream& operator<<(ostream& , Animation&);
};
```

```
// ass3.cpp
#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC     // need this to get the line identification
//_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF); // in main, after local declarations
//NB must be in debug build
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <string>
#include <vector>
#include <forward_list>
using namespace std;

#include "Display.h"
#include "SystemMemoryDisplay.h"
#include "GPUMemoryDisplay.h"
#include "Frame.h"
#include "Animation.h"

bool running = true;

int main(void)
{
          char selection;
          bool running = true;
          Animation A("A");
          _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF); // in main, after local declarations

          while (running)
          {
                    cout<<    "MENU\n 1. Insert a Frame\n 2. Delete all the Frames\n 3. Run the Animation\n 4. Quit\n"<<endl;
                    fflush(stdin);
                    cin>>selection;

                    switch (selection)
                    {
                    case '1':
                              A.InsertFrame();
                              break;
                    case '2':
                              A.DeleteFrames();
                              break;
                    case '3':
                              cout << A << endl;
                              break;
                    case '4':
                              running = false;
                              break;
                    default:
                              break;
                    }
          }

          return 0;
```

```
}
```

Example Output

```
MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Frame_1
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 2
Please enter pixel x-width for Display #0 pixel_x:1
Please enter pixel y-width for Display #0 pixel_y:1
Please enter the duration for this Display: 1
Please enter the name for this Display: Display_1
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 1

Please enter pixel x-width for Display #1 pixel_x:16
Please enter pixel y-width for Display #1 pixel_y:64
Please enter the duration for this Display: 2
Please enter the name for this Display: Display_2
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 2
Please enter the file name of the associated GPU Shader: Shader_1

This is the first Frame in the list

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Frame_2
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 1
Please enter pixel x-width for Display #0 pixel_x:128
Please enter pixel y-width for Display #0 pixel_y:128
Please enter the duration for this Display: 3
Please enter the name for this Display: Display_1
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 2
Please enter the file name of the associated GPU Shader: Shader_1


MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Frame_3
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 3
Please enter pixel x-width for Display #0 pixel_x:4
Please enter pixel y-width for Display #0 pixel_y:4
Please enter the duration for this Display: 1
Please enter the name for this Display: Display_1
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 1

Please enter pixel x-width for Display #1 pixel_x:2
Please enter pixel y-width for Display #1 pixel_y:2
Please enter the duration for this Display: 2
Please enter the name for this Display: Display_2
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 2
Please enter the file name of the associated GPU Shader: Shader_1
```

```
Please enter pixel x-width for Display #2 pixel_x:64
Please enter pixel y-width for Display #2 pixel_y:64
Please enter the duration for this Display: 4
Please enter the name for this Display: Display_3
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 2
Please enter the file name of the associated GPU Shader: Shader_2

There are 2 Frame(s) in the list
Please specify the position, between 0 and 1 to insert after : 0

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

3
Animation A
Run the Animation
Frame #0:       fileName = Frame_1
        Display #0: System Memory Display
        Display name = Display_1; pixel_x = 1, pixel_y = 1, duration =  1
        Counting the seconds for this Display: 1,
        Memory requirements = 8 bytes

        Display #1: GPU Memory Display. Shader = Shader_1
        Display name = Display_2; pixel_x = 16, pixel_y = 64, duration =  2
        Counting the seconds for this Display: 1, 2,
        Memory requirements = 4096 bytes

Frame #1:       fileName = Frame_3
        Display #0: System Memory Display
        Display name = Display_1; pixel_x = 4, pixel_y = 4, duration =  1
        Counting the seconds for this Display: 1,
        Memory requirements = 128 bytes

        Display #1: GPU Memory Display. Shader = Shader_1
        Display name = Display_2; pixel_x = 2, pixel_y = 2, duration =  2
        Counting the seconds for this Display: 1, 2,
        Memory requirements = 16 bytes

        Display #2: GPU Memory Display. Shader = Shader_2
        Display name = Display_3; pixel_x = 64, pixel_y = 64, duration =  4
        Counting the seconds for this Display: 1, 2, 3, 4,
        Memory requirements = 16384 bytes

Frame #2:       fileName = Frame_2
        Display #0: GPU Memory Display. Shader = Shader_1
        Display name = Display_1; pixel_x = 128, pixel_y = 128, duration =  3
        Counting the seconds for this Display: 1, 2, 3,
        Memory requirements = 65536 bytes


Output finished

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

1
Insert a Frame in the Animation
Please enter the Frame filename: Frame_4
Entering the Frame Displays (the sets of dimensions and durations)
Please enter the number of Displays: 1
Please enter pixel x-width for Display #0 pixel_x:1
Please enter pixel y-width for Display #0 pixel_y:1
Please enter the duration for this Display: 1
Please enter the name for this Display: Display_1
```

```
Please enter the type for this display (1 = SystemMemoryDisplay, 2 = GPUMemoryDisplay): 2
Please enter the file name of the associated GPU Shader: Shader_1

There are 3 Frame(s) in the list
Please specify the position, between 0 and 2 to insert after : 1

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit

3
Animation A
Run the Animation
Frame #0:       fileName = Frame_1
        Display #0: System Memory Display
        Display name = Display_1; pixel_x = 1, pixel_y = 1, duration =  1
        Counting the seconds for this Display: 1,
        Memory requirements = 8 bytes

        Display #1: GPU Memory Display. Shader = Shader_1
        Display name = Display_2; pixel_x = 16, pixel_y = 64, duration =  2
        Counting the seconds for this Display: 1, 2,
        Memory requirements = 4096 bytes

Frame #1:       fileName = Frame_3
        Display #0: System Memory Display
        Display name = Display_1; pixel_x = 4, pixel_y = 4, duration =  1
        Counting the seconds for this Display: 1,
        Memory requirements = 128 bytes

        Display #1: GPU Memory Display. Shader = Shader_1
        Display name = Display_2; pixel_x = 2, pixel_y = 2, duration =  2
        Counting the seconds for this Display: 1, 2,
        Memory requirements = 16 bytes

        Display #2: GPU Memory Display. Shader = Shader_2
        Display name = Display_3; pixel_x = 64, pixel_y = 64, duration =  4
        Counting the seconds for this Display: 1, 2, 3, 4,
        Memory requirements = 16384 bytes

Frame #2:       fileName = Frame_4
        Display #0: GPU Memory Display. Shader = Shader_1
        Display name = Display_1; pixel_x = 1, pixel_y = 1, duration =  1
        Counting the seconds for this Display: 1,
        Memory requirements = 4 bytes

Frame #3:       fileName = Frame_2
        Display #0: GPU Memory Display. Shader = Shader_1
        Display name = Display_1; pixel_x = 128, pixel_y = 128, duration =  3
        Counting the seconds for this Display: 1, 2, 3,
        Memory requirements = 65536 bytes


Output finished

MENU
 1. Insert a Frame
 2. Delete all the Frames
 3. Run the Animation
 4. Quit
```